Revisiting Continuous Deployment Maturity : A Two-Year Perspective

Mäkinen, Simo

ACM
2019-04-08

Mäkinen , S , Lehtonen , T , Kilamo , T , Puonti , M , Mikkonen , T & Männistö , T 2019 , Revisiting Continuous Deployment Maturity : A Two-Year Perspective . in Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing . SAC '19 , ACM , New York, NY, USA , pp. 1810-1817 , ACM/SIGAPP Symposium on Applied Computing , Limassol , Cyprus , 08/04/2019 . https://doi.org/10.1145/3297280.3297458

http://hdl.handle.net/10138/321583
https://doi.org/10.1145/3297280.3297458

acceptedVersion

Revisiting Continuous Deployment Maturity: A Two-Year Perspective

Simo Mäkinen
University of Helsinki
Helsinki, Finland
simo.v.makinen@helsinki.fi

Timo Lehtonen
Solita Ltd
Tampere, Finland
timo.lehtonen@solita.fi

Terhi Kilamo
Tampere University of Technology
Tampere, Finland
terhi.kilamo@tut.fi

Mikko Puonti
Solita Ltd
Tampere, Finland
mikko.puonti@solita.fi

Tommi Mikkonen
University of Helsinki
Helsinki, Finland
tommi.mikkonen@helsinki.fi

Tomi Männistö
University of Helsinki
Helsinki, Finland
tomi.mannisto@helsinki.fi

ABSTRACT

Background: Achieving a steady stream of small releases and employing practices such as continuous deployment requires maturity in company processes. Maturity models provide one approach for companies to pinpoint areas of improvement by providing a position and hints to reflect on. Incorporating maturity models with agile software development and continuous deployment has its challenges, though. **Aims:** The focus of the study is in understanding the evolution of software processes towards continuous deployment in an industry organization over time when a maturity model is used as a yardstick in evaluation. **Method:** An embedded case study by design, the study utilizes and replicates a survey on the state of software projects in a large Finnish software company, Solita. The survey was initially conducted in 2015 with responses from 35 projects and now replicated in 2017 with responses from 43 projects. Both quantitative and qualitative approaches for survey responses are used in the analysis. **Results:** Maturity of software processes in the case company show improvement in deployment and in monitoring, albeit short of statistical significance. Technological advances in the application of cloud computing have likely spurred development in these areas. Capability in processes related to test automation and quality has not changed much in two years. **Conclusions:** Maintaining maturity in software processes requires constant attention as impressions on process quality can gradually diminish. Projects which are built on a compatible technology stack have a greater chance in achieving continuous deployment and thus being more mature. Customer preferences also make a difference in the ability to reach certain maturity levels.

CCS CONCEPTS

• **General and reference** → **Surveys and overviews**; **Measurement**; • **Software and its engineering**;

SAC '19, April 8–12, 2019, Limassol, Cyprus

ACM ISBN 978-1-4503-5933-7/19/04…$15.00
https://doi.org/10.1145/3297280.3297458

1 INTRODUCTION

Modern software companies aim to continuously shorten their development and deployment cycles. The goal is to release new features as soon as they are ready instead of making large releases of many features developed over several months. This practice, called continuous deployment, CD [5], requires a mature enough development and deployment infrastructure – so-called *deployment pipeline* [7] – from code to delivery and deployment, matching practices [10], as well as organizational maturity [15].

In general, maturity models (e.g. [4, 12, 13]) have been introduced to help companies to assess the maturity of their software practices and to pinpoint areas where improvements are needed. While these models provide a position and hints for companies to reflect upon, acting on the results and tracking the progress of process changes towards the desired maturity is far from easy, especially in large organizations. In fact, there is little evidence on what happens in reality over time when a survey for assessing software development process maturity has been employed. Using maturity models should over time impact company processes towards sufficient development and deployment practices thus further improving the company's organizational maturity.

This paper investigates the use of a self-developed maturity model for the specific needs of a Finnish software service company Solita Ltd. over a two-year time window. The company focuses on software development as a service, data intensive projects (including both data warehousing and business intelligence and data analysis), and data analytics for customer projects. We report results from a survey study on a continuous deployment maturity model survey conducted first in 2015 [15] and now as a follow-up in 2017 for over 40 software projects in total. The goal is to study how development processes have evolved over time. The results indicate that on average the maturity levels of processes have mainly remained on the same level with a few clear exceptions. In some cases, while the processes have remained the same, the perception of maturity has even decreased. This points out the need for constant evolution and improvement of the infrastructure and processes. Maturity models and surveys can help to give a sense of direction and trigger the needed changes.

The rest of the paper is structured as follows. Section 2 provides background and motivation for this work. Section 3 introduces the research approach, and introduces the case company. Section 4

presents the results. Section 5 discusses the findings, and Section 6 draws some final conclusions.

2 BACKGROUND AND MOTIVATION

Maturity models for software development are available both as general purpose models and as models self-developed by software development organizations. The best known model is Capability Maturity Model Integration (CMMI) [14] developed by the Software Engineering Institute (SEI) in Carnegie Mellon University in collaboration with both the industry and the US Department of Defense. The CMMI framework uses five maturity levels: 1) initial, 2) managed, 3) defined, 4) quantitatively managed, and finally, 5) optimizing. These levels are based on artifacts and practices that have been considered essential for satisfying certain needs in terms of processes and activities [14]. Unfortunately, CMMI has been somewhat incompatible with agile practices until version 1.3, and to some degree this history still affects its role in industry. Moreover, the model is somewhat coarse-grained with its five levels, leaving various details undiscovered and emphasizing that all the characteristics of levels are met.

Ericsson's model by Rehn et al [13], Eficode's [2] DevOps Maturity Model, and Continuous Deployment Maturity Model [4] by Forrester Consulting all represent models proposed by software companies themselves. The Continuous Deployment Maturity Model derives from CMMI and each model utilizes a similar tiered approach for maturity assessment. A somewhat different approach proposed by Dzone research [1] is utilized in their continuous delivery maturity checklist. The checklist uses checkboxes that can be ticked or left blank for a particular activity depending whether the activity is in use or not. However, the checklists too uses a five-tiered categorization for maturity ranging from baseline to expert levels.

The Stairway to Heaven (StH) model for continuous deployment by Olsson et al. [10, 11] depicts a five-staged path from traditional development to experimentation and innovation. StH is further extended by Karvonen et al. [8] with practical viewpoints to the stages. In [9] Karvonen et al. construct CRUSOE, a framework for the analysis of continuous software engineering approaches in software intensive projects. It highlights the interdependencies of company internal and ecosystem architecture, strategy and organizing.

Helgesson et al. [6] present a mapping study on how maturity models have been evaluated. They show that two thirds of maturity model evaluation is done on self-developed models. They also categorize evaluations into a three level framework. Our aim here is to focus on the maturity evolution in a software service company aiming for maturity improvement via a self-developed maturity model on project maturity, not to evaluate the model itself.

Despite the large number of different maturity models, there is little evidence on their use in continuous software engineering. Furthermore, there is little empirical evidence, apart from anecdotal observations. However, it has been suggested that maturity models might not be a good fit for agile software development projects where team capabilities and flexible work habits are as important as any fixed process [3]. Similarly, existing maturity models are predominantly geared toward software product development.

All the above points imply that software project companies need to self-develop models adjusted to a development context where there are several independent projects running in parallel within the company with differences in their maturity and capabilities. Still, understanding models is beneficial for companies, whose business relies on the maturity to set up and maintain projects where continuous delivery is performed. Furthermore, investigating project maturity can shed light on the overall company maturity.

3 RESEARCH APPROACH

The study presented focuses on the use of a maturity model self-developed for a software project company's needs. The study spans over a two-year time window in a Finnish software service company Solita Ltd. The research question this survey study aims to answer is:

- RQ: How has continuous deployment maturity evolved in a software development organization over two years?

3.1 Case Definition

The study was conducted at Solita Ltd.[1], a Finnish software service company with more than 600 employees. Solita's core competence is delivering projects as a service. This means each project is tailored – at least in part – to meet the needs of each customer as they are working on custom software systems. There is also significant variance in the size of the projects, ranging from two to hundreds of man-months per project.

In 2015 Solita aimed to evaluate their in-house continuous delivery and deployment capabilities. In order to do this, Solita explored various existing maturity frameworks discussed in Section 2. However, the existing metrics and models fail to address two aspects specific to Solita's market sector. Firstly, product development differs from Solita's core business of project as a service, in particular, when it comes to return on investment (ROI) characteristics. Secondly, the size of the project affects the capability to establish a reasonable deployment pipeline and CD practices. Where large projects require significant effort to setup CD pipeline and practices, it may not be feasible to small projects due to other constraints such as staffing, time and budget constraints. In 2015, an in-house maturity model, the *Solita Test* [15], was developed to assess the CD maturity in Solita's development and data warehousing projects. The model evaluates the maturity of the project in five maturity categories (Figure 1). These included development process areas *Test automation* (TA), *Quality* (Q), *Build and Deployment* (BD) and *Running and monitoring* (RM) together with *Typical lead time* (LT). Each category has five maturity levels ranging from basic project requirements to full CD.

3.2 Method

As the study focused on investigating practices relating to the continuous deployment phenomenon and development process maturity in the real world and in an industrial context, the research approach is a case study. More specifically, the study design corresponds to an embedded case study design [16] with multiple units of analysis. The main unit of analysis is the organization and its

[1] http://www.solita.fi

	Test automation	Quality	Build & deployment	Running & monitoring	Typical lead time
5 ★★★★★	• HA tests • Tests run in parallel	• A/B testing • User analytics direct QA measures	• Continuous delivery to production • Build promotion procedures • Zero-downtime deployment • Scripted, one click rollback (incl db)	• Application performance monitoring • Automated application usage auditing • Radiator for application usage (business value)	**< 1** day
4 ★★★★	• Automated test reports and trends • Automated performance tests • Automated security tests	• Zero defect policy • Root-cause analysis • Fail proofing	• Continuous deployment to customer QA environment • Monitored Customer QA environment • Environment configuration in a separate version control repository	• Application team rutinely inspects logs • Automated alerts based on application logs • Server performance monitoring	**1-5** days
3 ★★★	• End-to-end tests • Browser based end-to-end tests	• Exploratory testing methods • Prototype based design verification • Go & See	• Build radiator • Scripted deployments to QA • Configuration is documented • Controlled DB migrations	• Client side error logging to server • Team has access to production logs • User statistics collection and analytics	**1-2** weeks
2 ★★	• Integration tests	• QA environment matches production • QA process is well documented	• Separate build server • CI build process triggered automatically by new commits	• User action logging (audit trail)	**2-4** weeks
1 ★	• Unit tests	• Customer QA, acceptance test step before production	• Version control	• Basic logging: error, access	**> 1** month

Customer = Client company that employs a Services company to build a solution
Customer's customer = Usually the actual end user of the solution
Deployment = Fully automated delivery with no human intervention
Delivery = Automated delivery with human triggered step(s)

Figure 1: 2015 CD maturity scale defined for the company's development context.

processes. On the subunit level, the embedded units under study are individual projects and further centering on individual respondents when asking them to express their views.

The primary research method use in the study was a survey based on a previously employed maturity test, the first Solita Test [15]. The replicated, extended survey was directed at Solita's project development teams. The survey was conducted through an online questionnaire. A link to the questionnaire was sent to an internal mailing list reaching all development teams at Solita. The questionnaire was published in May, 2017. It was open for two weeks.

In the questionnaire, respondents were asked to judge the maturity level of their project in five different development process areas: *test automation*, *quality*, *build & deployment*, *running & monitoring*, and *security*. The four areas (TA, Q, BD, RM) were already used in the 2015 study with *security* (S) added as a new process area for this study. For each area, the respondents could choose a maturity level ranging from 0 to 5 where the number 0 represented the lowest maturity level and number 5 represented the highest maturity level in the specific process area. Maturity levels were assessed for the project's current and target state. A supportive maturity matrix was given to the respondents as an external resource to guide the selection of the maturity levels.

Besides collecting data for the maturity levels, respondents were asked what was the *lead time* in their project, i.e., how long does it typically take for software changes made in the project to propagate to the production environment. Possible options for lead time were fixed to the following: during the same day, 1–5 days, 1–2 weeks, 2–4 weeks and last, more than a month.

At the end of the questionnaire, there was a free form feedback section. In this section, respondents could give feedback about the survey itself and reflect on areas of improvement for the survey. In addition, the questionnaire had fields for background questions such as team size, used technologies and customer domain.

Out of approximately 150–200 projects, there was response from the representatives of 43 projects (an increase from 35 in the 2015 study). Thus, the response rate was approximately one fourth of active projects in the case company.

Table 1 shows more detailed information on the survey responses. The projects were divided into groups by the amount of persons participating the project by using the following categories: small projects (1–3 persons), medium projects (4–9 persons) and large

Table 1: Survey responses per project size and type

Year	N	Software-intensive	Data-intensive	Size: S	Size: M	Size: L
2015	35	29	6	7	23	5
2017	43	35	8	18	21	4

projects (10 or more persons). We consider the medium-sized projects to be the most relevant group from the CD maturity point of view for two reasons. First, they represent projects in active development lifecycle. In this phase of the project, the foundations of continuous delivery tools and practices are created. Inactive projects in maintenance phase often have less people working in them and investments to CD tools and practices have already been done. Second, projects in active development phase that have a larger group of people working together benefit from CD tools and practices. For instance, short cycle feedback produced by the servers in the deployment pipeline, is useful for such projects. An automatically triggered CI build process together with automated database migrations, end-to-end testing and build radiator makes the cooperation of the team more efficient. Moreover, we leave out large projects (with 10 persons or more) and concentrate into medium-sized projects in order to find a set of projects that have similar kind of characteristics.

The responses from the projects were categorized into two groups: data-intensive and other. The division into these categories was made due to the case company participating a research program[2]. In it, the case company had set a goal to improve delivery capabilities especially for data-intensive solutions. Term data-intensive in this case refers to data-centric software solutions that concentrate into data warehousing and data analysis. The goal was to drastically improve delivery capabilities for them. The company sought to achieve a full 100 percent improvement in productivity and halve lead times. The goal was to double the number of high quality solutions delivered in a given time frame. Given the goal, data-intensive projects pose a special focus of interest in the analysis of the results.

In addition to the maturity level data, the free form questions were analyzed in order to gain insight on the participants wider views on the model and its use. The use of different technologies was also included in the free form questionnaire. The answers were first annotated by one of the researchers and then the coding was checked and augmented by another researcher. The second researched further triangulated the results with the results of the 2015 semi-structured interview results.

4 RESULTS

The results are presented from five points of view for different units of analysis. First, we demonstrate the overall evolution of continuous deployment maturity in the case company. The data set in this case consists of all the projects that had responses for the survey in 2015 and 2017. Second, we present tracking data of 10 projects that had survey responses both in 2015 and in 2017, shifting the unit of analysis to the project level. Third, we evaluate the evolution of lead times in the case projects. Fourth, we estimate the technological advancements made in the company based on utilized technologies for the projects. Finally, we analyze the open survey feedback from the perspective of the respondents.

[2]http://n4s.fi

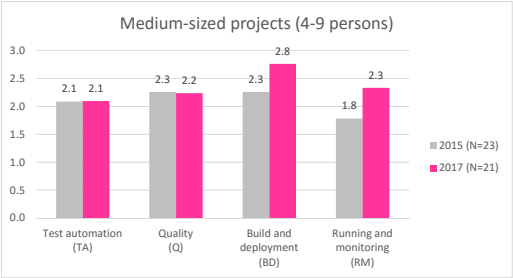

Figure 2: Mean of CD maturity dimensions for medium-sized projects (4–9 persons) in 2015 and 2017.

4.1 Overall Evolution of Continuous Deployment Maturity

The overall evolution of continuous deployment maturity of the case company is presented in Figure 2. The results consist of survey data of medium-sized projects in 2015 and 2017.

According to the data, the evolution of the dimensions is two-fold. The average of test automation and quality has stayed at the same level. The average of dimension *Build and deployment* has increased from 2.3 to 2.8. Accordingly, *Running and monitoring* has increased from 1.8 to level 2.3.

To investigate if the difference is statistically significant, we applied the Mann-Whitney U-test to the dataset. The test does not take the distribution of the dataset into account and is thus suitable in this case. For running and monitoring, the change is only statistically significant with p-value 0.1 so there are some signs of a positive trend. The size of the data sample is rather small which increases the p-value and brings uncertainty to statistical significance. We conclude that the change for dimensions *BD* and *RM* is not statistically significant.

4.2 Longitudinal Tracking of Single Projects

Ten of the surveyed projects in 2017 were ongoing projects which had survey replies also from the 2015 survey. Table 2 illustrates the reported maturity levels (see Figure 1) of each project, with the difference to the previous survey in parentheses. Positive values mean that maturity has improved in a dimension compared to 2015.

It is noteworthy that the evolution of CD maturity dimensions for some of the projects is negative. For instance, the quality dimension of project *App4* has dropped from level 4 to 2. On the other hand, for

Table 2: Continuous deployment maturity of tracked projects in 2017, difference to 2015 responses in parentheses

Project	TA	Q	BD	RM	S	Lead Time	TA Target	Q Target	BD Target	RM Target	S Target	Lead Time Target
App6	2 (0)	4 (+1)	4 (+1)	2 (+1)	2	1 (-2)	4 (0)	4 (0)	4 (0)	4 (0)	4	3 (-2)
App22	3 (+1)	3 (+1)	2 (0)	2 (0)	4	1 (0)	3 (0)	4 (+1)	4 (+1)	3 (0)	4	1 (0)
App11	4 (0)	2 (+1)	3 (0)	2 (0)	0	1 (0)	4 (0)	2 (-1)	4 (0)	2 (-2)	3	1 (0)
DW1	1 (-1)	3 (+1)	2 (0)	4 (+1)	1	2 (-2)	4 (0)	4 (0)	4 (+1)	5 (0)	3	2 (-3)
App2	3 (+1)	3 (0)	2 (0)	2 (-1)	1	1 (0)	4 (0)	3 (-1)	4 (0)	4 (-1)	3	2 (0)
App14	2 (-1)	0 (-1)	3 (+1)	3 (+1)	1	3 (0)	2 (-2)	0 (-4)	4 (-1)	4 (-1)	2	3 (-2)
App4	4 (+1)	2 (-2)	3 (0)	4 (0)	1	1 (-1)	4 (0)	3 (-1)	5 (+1)	4 (-1)	2	1 (-1)
App15	3 (0)	2 (0)	2 (0)	1 (-1)	0	2 (0)	3 (-1)	2 (-2)	3 (0)	2 (-2)	2	2 (0)
App21	2 (-1)	3 (0)	3 (0)	2 (0)	0	1 (0)	3 (-1)	3 (-1)	4 (0)	4 (0)	3	2 (0)
App19	0 (0)	1 (-1)	1 (0)	1 (0)	2	2 (0)	0 (-3)	1 (-3)	2 (0)	3 (0)	2	2 (0)
Mean	**2.4 (0)**	**2.3 (0)**	**2.5 (+0.2)**	**2.3 (+0.1)**	**1.2**	**1.5** (-0.5)	**3.1 (-0.7)**	**2.6 (-1.2)**	**3.8 (+0.2)**	**3.5 (-0.6)**	**2.8**	**1.9 (-0.8)**

instance, project *App6* has improved all dimensions except testing. Moreover, the setting of future goals has become more moderate which can be seen in the overall drop in the target levels the teams aim to achieve. The mean of dimension *TA* was 3.8 in 2015 and 3.1 in 2017 and *Q* 3.8 and 2.6 accordingly for the target levels. In dimensions *BD* and *RM* the goal has stayed on a higher level.

4.3 Analysis of Lead Times

Lead time represents the typical time it takes in a project to push out new releases of project deliverables. As such, there have been slight changes in the lead times in two years.

Looking at the results for the whole company across all the projects from which there were survey responses, on average there has been a minor shift towards releases between one and two weeks as the average has changed from 2.1 to 2.2. Surprisingly, the most common value, the mode, marks that the majority of all the projects in 2017 had a typical lead time of over one month. In 2015, the mode was 2–4 weeks for the lead time which is somewhat shorter. The median has stayed the same at 2–4 weeks for 2015 and 2017.

On the more rapid end of the lead time scale, the responses show that deployment during the same day or even during the same week is comparatively rare and projects choose to deploy less often. From all the responses, there were only two projects in 2017 where the lead time was a day or shorter. Still, this is an improvement over 2015, when there were no projects that could deploy during the same day. Making releases and deploying changes inside a typical work week is not that common either, since for both 2015 and 2017 there were five projects that could release and deploy so often.

It seems that few projects even *want* to cut their lead times so that they could deploy within the same day. Only three projects set same day deployments as their target state in 2017 compared to four projects in 2015. For most projects, it is ok if the changes go out within the week or maybe in two weeks. The results indicate, however, that it has become more attractive to wish for releases in a week or two. This is a clear shift in attitudes towards faster deployment in general although not evident in the tracked projects.

Comments from the respondents suggest that it is not completely straightforward to define or select the typical lead time for a project. Because small fixes can be done in a day or two, their lead time is shorter than for other features which might take the whole cycle to develop properly.

4.4 Advances in Technology

Novel tools and technologies can spur continuous deployment maturity by making the life of a developer easier or enabling different workflows that were not previously possible. Responses to the Solita survey both in 2015 and 2017 paint a picture to technological advancements in the company. From the free form technological descriptions given as part of the survey responses, around a 100 distinct technologies or tools could be identified in 2015 and closer to 160 in 2017. The names of the technologies and tools were harmonized by grouping names with slightly different spelling together. While the most frequently mentioned technologies are reliable workhorses that get the job done year after year and have remained largely the same, it seems there are new emerging technologies that have the potential to improve continuous deployment maturity.

The survey responses from 2017 provides an overview of the used technologies as illustrated in Figure 3. It would be fair to say that Java is the technological foundation for many projects since Java and Spring Boot were among the ten most frequently mentioned technological terms in the responses. Many projects seem to target web platforms as Angular and the React JavaScript library appear also frequently in the comments. For database solutions, Oracle is a key technology based on term frequency but it is closely followed in popularity by the object-relational mapping framework Hibernate with PostgreSQL having several mentions as well. Integration projects seem to have some prominence, too, with the appearance of the enterprise service bus technology Mule ESB in the responses. Many other technologies and tools were mentioned but they were not among the top 10 list of most frequently mentioned technologies and tools.

Taking a closer look at the technologies and tools in 2017, reveals, however, signs of change visible in the technology stack of Solita. During two years, the company has introduced a whole suite of Amazon's web services in its projects. Infrastructure for projects is being provided by Elastic Compute Cloud (EC2), partly utilizing container services of Amazon and on-demand computing with Lambda nodes. Besides taking advantage of computing power, content storage is in certain cases handled through similar services both for static files and database content, used especially for data warehouses. Monitoring of infrastructure and the current state of running services is also potentially facilitated through Amazon's Elasticsearch as is deployment of applications through CloudFormation. These services might help projects reach higher maturity levels

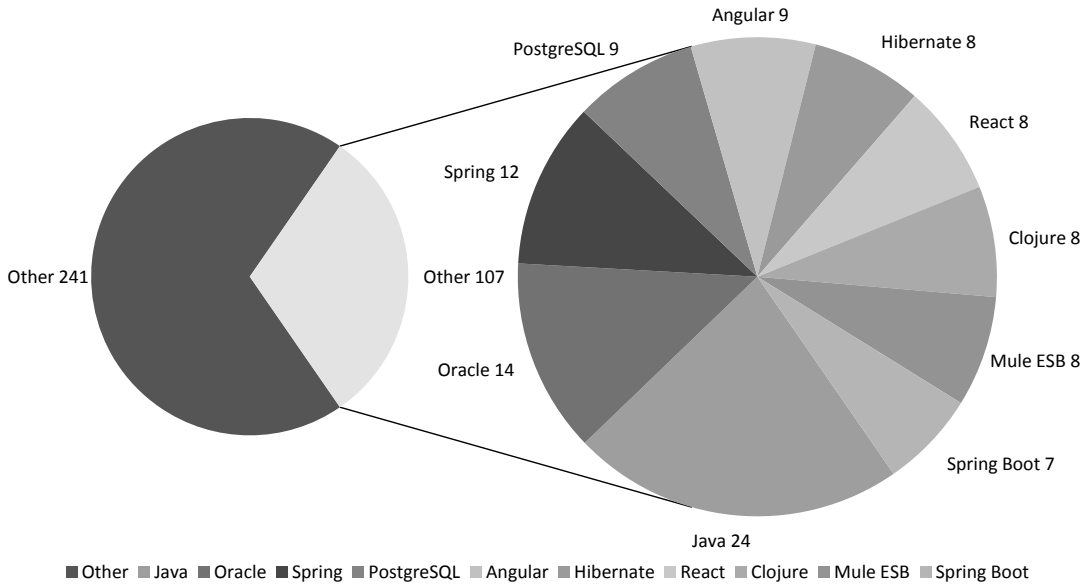

Figure 3: Term frequency in technology comments in 2017, 10 most frequent technologies

for build and deployment, and for running and monitoring which require deployments with little downtime and real-time monitoring of environments with alerts.

4.5 Open Survey Feedback

As a part of the survey, the respondents had the possibility to give general feedback about the survey and about the application of maturity models for assessing current project practices and activities. In 2017, out of 43 project correspondents, 25 had commented on the survey. The 31 feedback comments out of 35 projects from the 2015 maturity survey were analyzed as well in addition to the more recent survey. In total, 56 comments were analyzed.

In total, there were 11 comments addressing the survey itself, out of which all but one of which regarded the survey positively. The respondents felt that figuring out responses to the survey and fitting the maturity model to project practices had its uses. At least seven feedback comments from 2017 and 2015 shared the observation that the survey provoked thoughts among the respondents and forced them to critically consider project practices. The maturity model was seen to give a sense of direction to improvement endeavors. Some concerns (four comments) were raised regarding difficulties in knowing how to answer or interpret the teams maturity.

Motivational factors were mentioned in 9 comments. They ranged from curiosity to see how the team compares to others' results to reasons why no improvement is sought (two comments). Five respondents wanted to get their hands on the company-wide results so that they could compare their own results and project practices with the results of the others. In addition, one felt that the survey showed positive surprise on the perceived maturity. One comment mentioned the possibility to reward further improvements.

An overarching concern related to the application of the maturity model was the rough division of the maturity levels in the process areas. There were also comments about the order of the maturity levels. A higher maturity level was for example considered something a more immature project should already have. At times, there was not a perfect fit between the maturity levels and actual project practices. A project might have a particular practice in use from a higher maturity level but lack some other practice from a lower level. Selecting the correct level was not easy. Practices were not seen to be as linear as depicted in the maturity model. The fact that practices do not follow a linear curve from the least to the most advanced was highlighted in 16 feedback answers. A solution proposed by some of the respondents included having a distinct set of practices which they could tick instead of having to assess and choose the level as such.

Solita orchestrates projects from different domains and for different purposes which were in certain cases deemed incompatible with the general maturity model. While the practices in the maturity model were seen to reflect some of the best practices in a standard web application development project, the model was not seen to completely fit business intelligence projects or projects that are not deployed in such a direct manner like desktop application projects.

Project maturity is not always in the hands of the project team, either. Although the project team might have all the knowledge to take practices from a higher maturity level in use, the project customer might restrict their freedom to apply specific practices. More than a few respondents mentioned that project maturity also depends on who is in charge. Acceptance tests are hard to implement if the customer does not have the habit of conducting them. Customers can also restrict visibility and access to their production systems which can make it difficult to reach maturity target states in projects. It is up to the customer to decide what activity has the highest priority, and overall process improvement is not always on the top of the list. These limitations are in line with the interview results of the study conducted together with the 2015 survey [15]. Transparency issues such as access to customer environments and

data, and sufficient customer feedback, communication issues with the customer, cultural issues such as customer's support for changing development practices and technical issues such as restrictions in the development environment as well as resource issues match the issues impacting improvement found in the 2015 study.

According to the feedback comments, also the project lifecycle has impact on the maturity of processes. In the early stages of a project, processes might not have reached their full maturity as processes are ramped up slowly from one release to the next. In such cases, the target state for the project as measured in the survey can be quite a lot different from the current state of the project. While young projects have a greater tendency towards eagerness in developing practices, for projects which are in the later lifecycle stages, the effect is the opposite. A respondent mentioned that for their project which is in the maintenance phase, there are not much resources available or interest in process improvement. The same goes for projects in the terminal phase that might be discontinued or barely kept alive in the foreseeable future.

5 DISCUSSION

The discussion of the results addresses the main research question,

- RQ: How has continuous deployment maturity evolved in a software development organization over two years?

To this end, responses from the previous survey are reflected to the more recent responses to describe evolution of project level practices. As part of the discussion, respondents' comments and conceptions regarding using maturity models are summarized.

Continuous deployment maturity at Solita appears to have slightly improved for its important medium-sized projects but the trajectories for maturity differ and not all kind of projects exhibit similar upward trends. Medium-sized projects are important to the company because they represent the most common type of actively developed projects in which advanced development practices could have the most beneficial effect. A further examination of the survey responses between 2015 and 2017 indicate that the two-year journey has not been the same for all the tracked projects, either, for which there was survey response data from both years.

Overall for the medium-sized projects, maturity has improved in build and deployment and running and monitoring although the difference was not deemed statistically significant. This means, for instance, that teams are in better control of their development environments with improved capability for deployment and environment configurations, and have more straightforward access to production environments and system logs. Although the improvement was smaller, a similar trend could be observed from the 10 tracked projects. However small, it is a step towards continuous deployment which cannot be achieved without such streamlined activities. Regarding test automation and quality, maturity has remained pretty much on the same level for medium-sized projects as well as the tracked projects. It is still rare to have a fully automated test pipeline within the company as requirements for higher test automation levels such as automated non-functional testing seem not to be met with some exceptions. When looking at the lead times, there is a slight change to the better on average lead times but the most common lead time is now in a slower category. Thus, the capability to release more often has not increased in medium-sized projects, and has decreased for the tracked projects. Perhaps lead time is rather specific to a project and its phase, dependent on customer preferences and its improvement is not entirely in the hands of developer teams, as indicated in the feedback comments.

Thinking about the future, projects are no longer as ambitious as before, either. Target levels have dropped, especially for testing and quality. Perhaps this means that the higher levels are no longer seen as desirable or otherwise attainable by current resources. The target for build and deployment has remained more constant but very few projects would be ready to shift to continuous delivery. All this shows that maintaining a certain level in continuous deployment maturity does not happen without investment and support. Organizations need to continuously put effort into maintaining the level of maturity, and to enabling improvement.

In comparison to other types of projects, data-intensive projects have progressed further, although the initial baseline in 2015 was far lower than for other projects. Maturity has improved in four categories, namely all areas except testing. Based on the improvements, direct changes in processes and tools can be observed. The positive change in deployment for data-intensive projects means that there is better support for scripted deployment to such test environments as required by the third tier in deployment. It is plausible that the improvement in deployment capability is to some extent associated with the introduction of new cloud infrastructure technologies like the Amazon web services stack mentioned in the free-form technology comments for 2017. Continuous deployment requires streamlined processes in the pipeline with related technologies and tools to support them. Testing remains as the only category for data-intensive projects with decreased maturity. This may be a consequence of concentrating on the improvement of deployment practices. As focus and thus staff and resources were placed on deployment practices, less focus was on improving or even maintaining testing practices. This indicates that if no attention is paid to maintaining and supporting current maturity levels, the practices can start to deteriorate.

From a technological perspective, there are certain predominant technologies favored by Solita both in 2015 and 2017. Many projects are based on web technologies, mainly Java which is apparently used for server-side solutions. In theory, such a technology stack could allow a high level of continuous deployment maturity since maintaining a steady flow of releases to web servers should be comparably easier than in other more restricted domains such as embedded systems. Testing, and particularly automated testing required by higher levels of the test automation category should also be possible in the web domain with known technologies as plenty of test frameworks are available for use. Perhaps the availability, or rather the lack, of suitable technologies and well-defined processes for testing explains the obvious gap between the standard web application and data-intensive projects in the survey responses and results. The appearance of new technologies in 2017 hints that development teams are allowed to experiment on suitable novel technologies and that good practices spread from team to team. Keeping the company culture experimental could be important in improving continuous deployment maturity over time by reducing the likelihood of staying stagnant and being stuck with old processes and technologies that might not allow the achievement of continuous deployment as well as could be possible.

Analysis on the survey feedback comments reveals that feedback has revolved around similar themes in 2015 as in 2017. For instance, on both accounts the respondents felt that it matters quite much in which lifecycle stage the project is in. Process maturity is not very high in projects which are not actively developed but only maintained. The domain also makes a difference. For some domains particular process areas are not relevant and thus a general maturity model can be seen unfit for use. Furthermore, a shared concern arising from the feedback comments was that views of continuous deployment maturity differ. There are various opinions about the exact order of particular practices in the maturity model and what should be included in the maturity requirements for each level. Nevertheless, the survey was regarded helpful in raising awareness of project practices in need of attention and pointing the way forward.

While the survey helps to highlight the required cultural changes, it is not alone sufficient to push for a change. More focus is needed on the issues impacting improvement. The customer plays a key role in enabling the application of improvements the teams are aiming for. This indicates the need to communicate the benefits of continuous deployment to the customer in order to on-board all stakeholders to achieving the improvement goals.

Threats to Validity. The main threat to the internal validity of the study is the respondent bias in filling the questionnaire. There is inevitable subjectivity in answering. The questionnaire came with a guide to describe each level in more detail to mitigate this threat. There are also differences in the respondents with only 10 projects having responses both from the 2015 and the 2017 surveys. For external validity the study context being limited to a single organization and the self-developed nature of the model pose a threat. While the results may not thus be fully applicable to other organizations the results are still interesting beyond the single-company scope. Still, there is a need for replication studies in other organizations. At current, the model, while developed taking the existing models into account, is targeted for a single software company's use which can harm the replicability of the study. The reliability of the study is tied to the response rate which was similar to the 2015 study and considered typical. The respondents chose to answer if they saw it fit as this was considered to give more reliable answers overall. In addition, change in the projects is inevitable due to the company's business model. Finally, while the 2015 study included semi-structured interviews of selected projects, this follow-up study chose to forgo them. Although the interviews might have strengthened the results with further triangulation, the focus here is on the organizational level more than the model and its perception. The results here are further strengthened by researcher triangulation of the free form answers by two researchers and combining these results with the results of the 2015 interview study.

6 CONCLUSIONS

In two years of software development, technology can advance swiftly as new development tools and practices emerge. Measuring the current technological state and the maturity of processes in a project can yield useful information for process improvement. As a software development company, Solita has employed a continuous deployment maturity survey in 2015 and 2017 to map its current technological state in projects.

Results from the survey indicate that for typical medium-sized projects the maturity for test automation and quality has stayed on the same level but has advanced for the areas of build and deployment, and running and monitoring. Data-intensive project have in particular improved their capability for build and deployment, possibly with the help of emerging technologies. The longitudinal analysis shows that for some projects, the processes might have been downgraded and thus the overall maturity has decreased in single cases. Evolution of projects does not always lead to increased maturity for various reasons. For instance, the typical lead times are in certain cases longer.

Surveys based on maturity models rely on the underlying maturity model construction. Feedback from the Solita survey shows that selecting the correct maturity level can be tricky. Opinions differ as to what practices should be included in the model and to which tier should the practices be tied. Still, conducting and answering to the survey can not only provide an overall picture of the company process but also give new insight to the respondents and act as a catalyst for software process improvement.

REFERENCES

[1] DZone [n. d.]. DZone Research CD Checklist. https://static.dzone.com/dz1/dz-files/CD_Checklist_0.pdf. Accessed: 2018-12-09.

[2] Eficode [n. d.]. DevOps Quick Guide. https://www.eficode.com/en/devopsguide. Accessed: 2018-12-09.

[3] Rafaela Mantovani Fontana, Isabela Mantovani Fontana, Paula Andrea da Rosa Garbuio, Sheila Reinehr, and Andreia Malucelli. 2014. Processes versus people: How should agile software development maturity be defined? *Journal of Systems and Software* 97 (2014), 140–155. https://doi.org/10.1016/j.jss.2014.07.030

[4] Forrester Research, Inc. 2013. Continuous Delivery: A Maturity Assesment Model. https://info.thoughtworks.com/Continuous-Delivery-Maturity-Model.html. Accessed: 2018-12-09.

[5] Martin Fowler. 2013. ContinuousDelivery. http://martinfowler.com/bliki/ContinuousDelivery.html. Accessed: 2018-12-09.

[6] Yeni Yuqin Li Helgesson, Martin Höst, and Kim Weyns. 2012. A review of methods for evaluation of maturity models for process improvement. *Journal of Software: Evolution and Process* 24, 4 (2012), 436–454. https://doi.org/10.1002/smr.560

[7] Jez Humble and David Farley. 2010. *Continuous delivery: reliable software releases through build, test, and deployment automation.* Pearson Education.

[8] Teemu Karvonen, Lucy Ellen Lwakatare, Tanja Sauvola, Jan Bosch, Helena Holmström Olsson, Pasi Kuvaja, and Markku Oivo. 2015. Hitting the Target: Practices for Moving Toward Innovation Experiment Systems. In *Software Business*. Springer, 117–131.

[9] Teemu Karvonen, Tanja Suomalainen, Marko Juntunen, Tanja Sauvola, Pasi Kuvaja, and Markku Oivo. 2016. The CRUSOE Framework: A Holistic Approach to Analysing Prerequisites for Continuous Software Engineering. In *Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17*. Springer, 643–661.

[10] Helena Holmström Olsson, Hiva Alahyari, and Jan Bosch. 2012. Climbing the "Stairway to Heaven" - A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software.. In *EUROMICRO-SEAA*, Vittorio Cortellessa, Henry Muccini, and Onur Demirörs (Eds.). IEEE Computer Society, 392–399.

[11] Helena Holmström Olsson, Jan Bosch, and Hiva Alahyari. 2013. Towards R&D as innovation experiment systems: A framework for moving beyond agile software development. In *Proceedings of the IASTED International Conference on Software Engineering, SE 2013*. 798–805.

[12] Mark Paulk. 1993. *Capability maturity model for software.* Wiley Online Library.

[13] Andreas Rehn, Tobias Palmborg, and Patrik Boström. 2013. Continuous Delivery Maturity Model. http://www.infoq.com/articles/Continuous-Delivery-Maturity-Model. Accessed: 2018-12-09.

[14] CMMI Product Team. 2002. Capability Maturity Model® Integration (CMMI), Version 1.1–Continuous Representation. (2002).

[15] Antti Virtanen, Kati Kuusinen, Marko Leppänen, Antti Luoto, Terhi Kilamo, and Tommi Mikkonen. 2017. On Continuous Deployment Maturity in Customer Projects. In *Proceedings of the Symposium on Applied Computing (SAC '17)*. ACM, New York, NY, USA, 1205–1212. https://doi.org/10.1145/3019612.3019777

[16] Robert K. Yin. 2014. *Case Study Research: Design and Methods* (5th ed.). SAGE Publications.